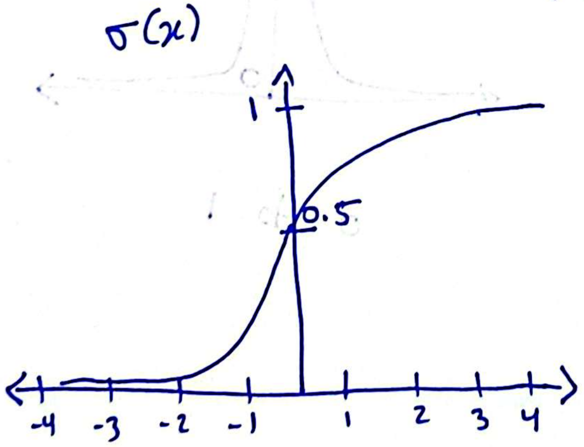
Activation Functions

Activation functions helps to determine the output of a neural network. These type of functions are attached to each neuron in the network, and determines whether it should be activated or not, based on whether each neuron's input is relevant for the model's prediction

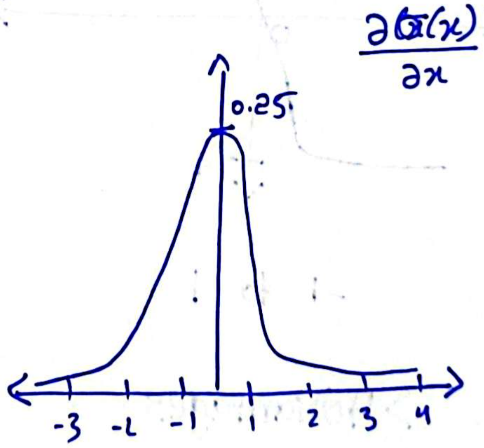
Activation functions also helps to normalize the output of each neuron to a specific range

→ Sigmoid Activation Function

$$\sigma(x) = \frac{1}{1+e^{-x}}$$, range : 0 to 1



0 to 1



0 to 0.25

> Advantages

① Smooth gradient, preventing "jumps" in output values

② Output values bound between 0 and 1, normalizing the output of each neuron

③ Clear predictions, i.e. very close to 1 or 0.

> Disadvantages

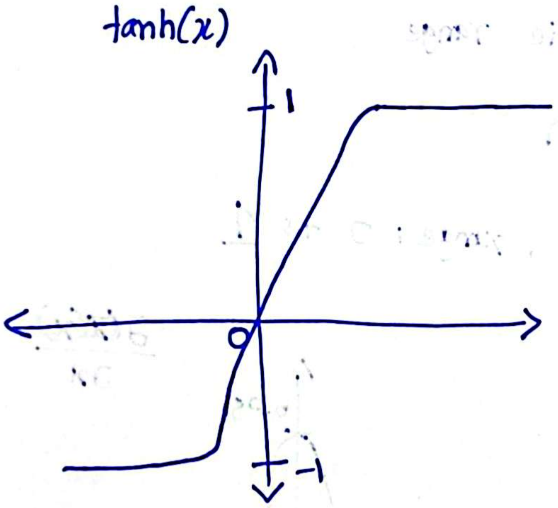
① Prone to gradient vanishing

② Function output is not zero-centered

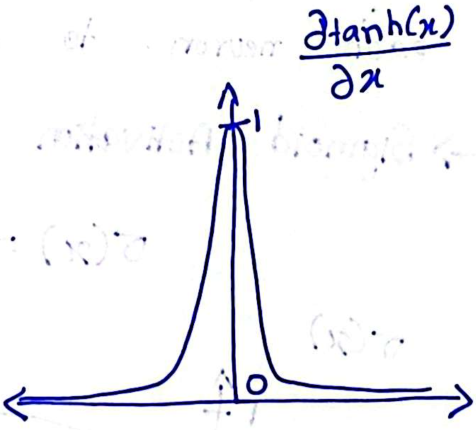
③ Power operations are relatively time consuming

⟶ Tanh Activation Function

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



-1 to 1



0 to 1

> Advantages

① Zero centric functions

② Modified version of sigmoid and comparatively better sigmoid function

> Disadvantages

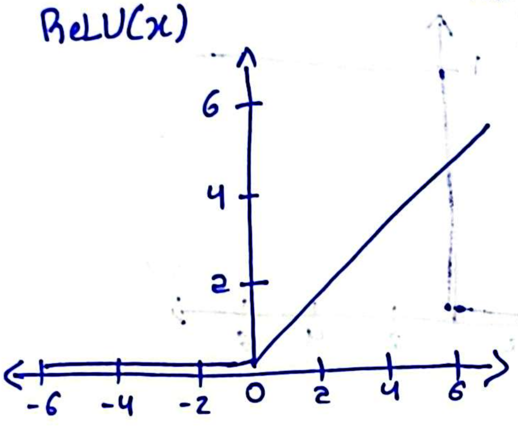
① Vanishing gradient issue for large layers

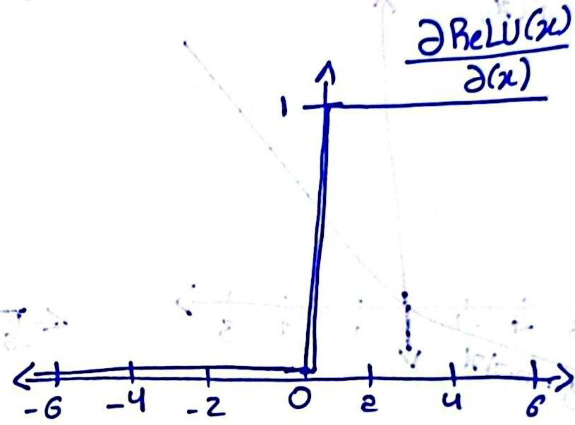
② Expensive computational power

→ ReLU Activation Function

$$x = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x \geq 0 \end{cases}$$

$$= \max(0, x)$$





0 to ∞

0 or 1

if ReLU output is 1 => weight updation

if ReLU output is 0 => Dead neuron

> Advantages

① Solving vanishing gradient problem

② It doesnot activate all the neurons

③ Computation is very fast.

> Disadvantages

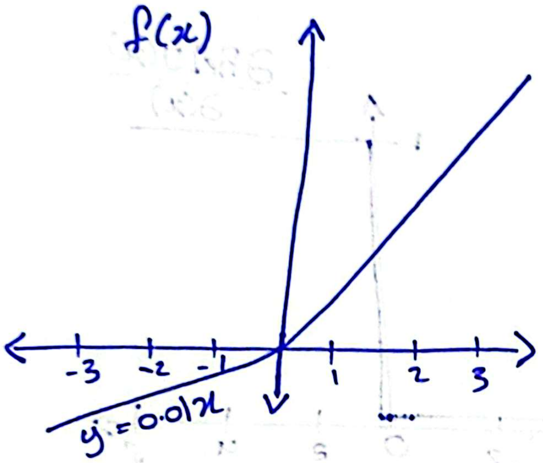
① ReLU function is not zero centric

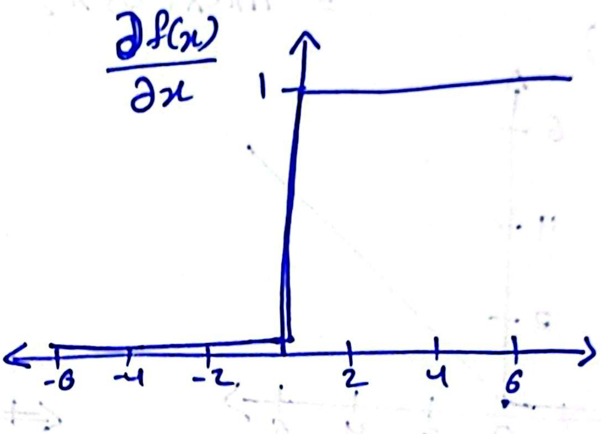
② Dead neurons as it deactivates the neurons for -ve value and do not provide -ve direction.

③ For -ve input it does not do anything, which is not good for learning of Neural Network.

→ Leaky ReLU

$$\begin{cases} 0.01x & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$





−∞ to +∞

> Advantages

① All the advantages of ReLU

② Overcome the dead neuron issue

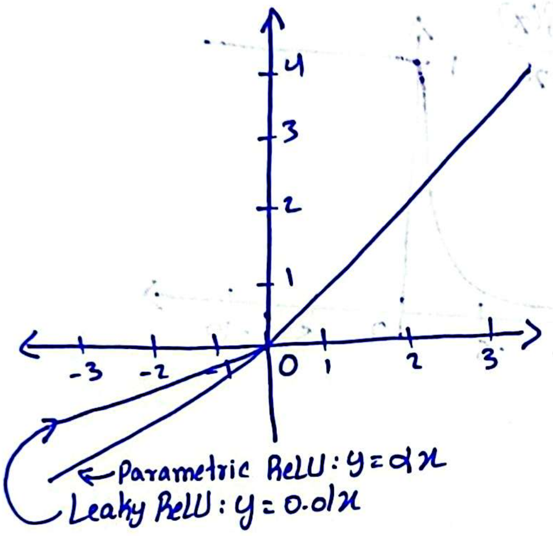
③ Less computational expensive compare to sigmoid and tanh

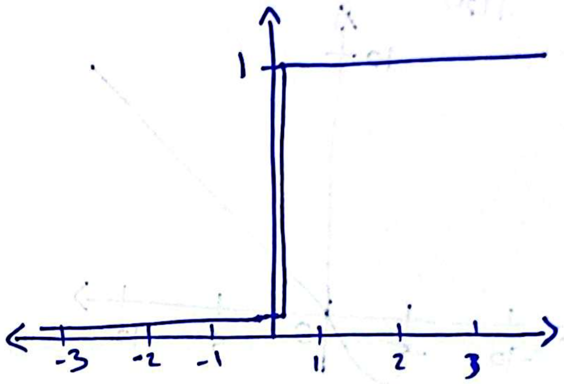
> Disadvantages

① It doesnot provide much learning for −ve ~~+ve~~ input as alpha value is constant

⟶ Parametric ReLU

$$\begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$





$-\infty$ to $+\infty$

> Advantages

① Slight advantage over leaky function as we can change learnable parameter alpha.

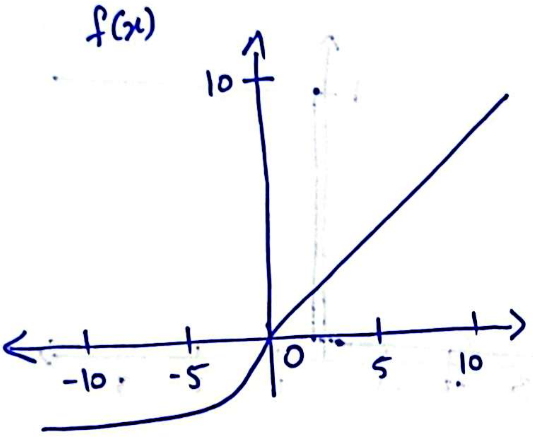
② Less expensive compare to exponential activation function.

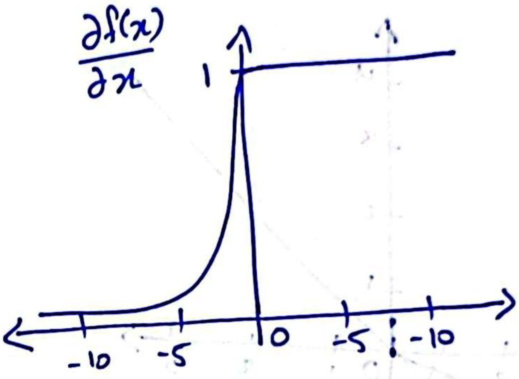
> Disadvantages

① Does not provide more learning for -ve dataset

→ Exponential Linear Unit (ELU)

$$\begin{cases} \alpha(e^{x}-1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$





$-\alpha$ to ∞

> Advantages

① It is zero-centric

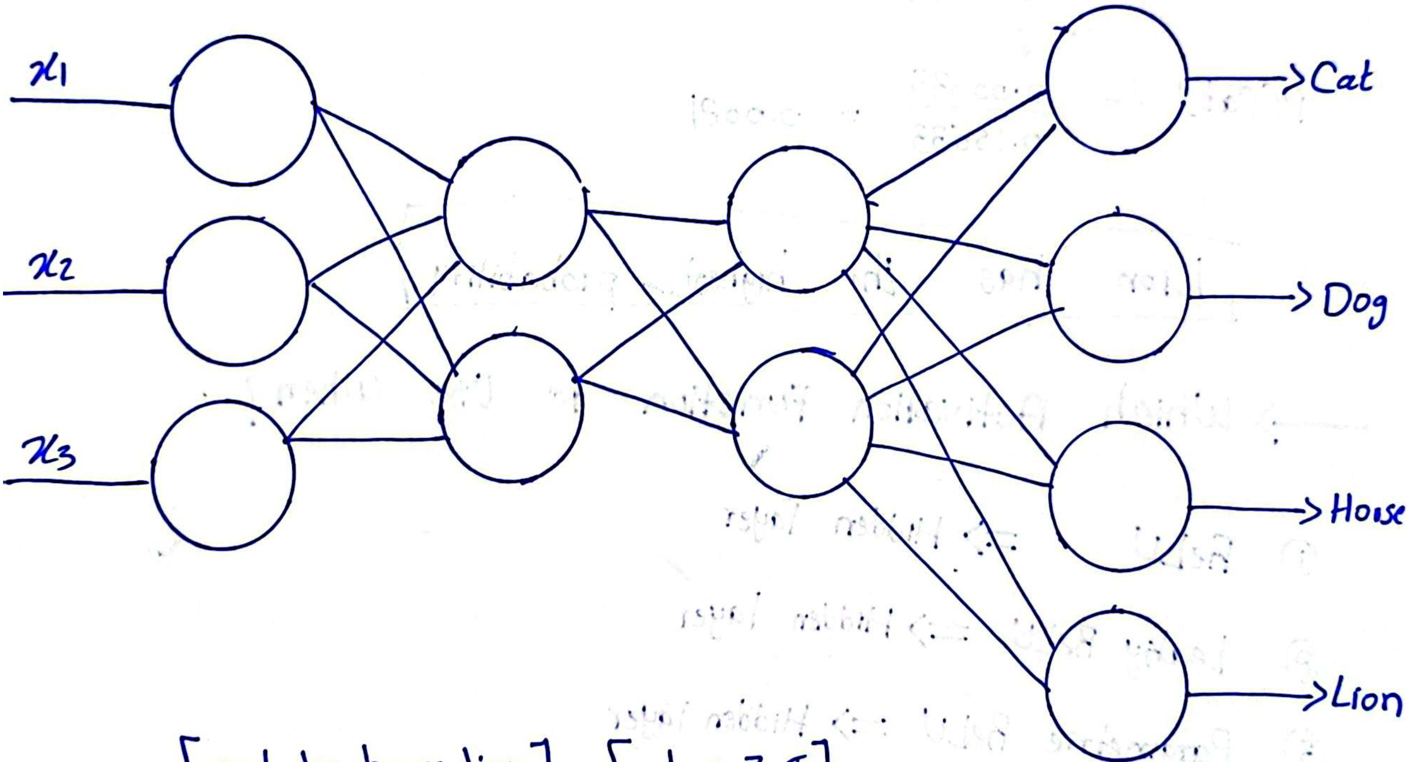
② Address issue of dying neuron

> Disadvantages

① Computationally expensive as exponential calculation is involved

→ Softmax Activation Function

Softmax activation function transforms the raw outputs of the neural network into a vector of probabilities, essentially a probability distribution over the input classes. Consider a multiclass classification problem with N classes.



[cat, dog, horse, lion] = [-1, 0, 3, 5]

$$\text{Softmax} = \frac{e^{y_i}}{\sum_{k=0}^{n} e^{y_k}}, \quad y_i = \text{output x.weight + bias.}$$

$$\text{Cat} = \frac{e^{-1}}{e^{-1+0+3+5}} = 0.003, \quad \text{Dog} = \frac{e^{0}}{e^{7}} = 0.0024$$

$$\text{Horse} = \frac{e^{3}}{e^{7}} = 0.0183, \quad \text{Lion} = \frac{e^{5}}{e^{7}} = 0.1353$$

$$P(\text{Lion}) = \frac{0.1353}{0.00033 + 0.0024 + 0.0183 + 0.1353} = 0.8654$$

$$P(\text{Horse}) = \frac{0.0183}{0.15633} = 0.11$$

$$P(\text{Dog}) = \frac{0.0024}{0.15633} = 0.015$$

$$P(\text{Cat}) = \frac{0.00033}{0.15633} = 0.0021$$

Lion has the highest probability

→ Which Activation Function to Use When?

1. ReLU => Hidden layer
2. Leaky ReLU => Hidden layer
3. Parametric ReLU => Hidden layer
4. Exponential Linear Unit => Hidden layer
5. Sigmoid => Binary classification, output layer
6. Tanh => Hidden layer, shallow networks
7. Softmax => Multiclass classification, output layer